DSA final

Team ID: Team 37

Student IDs: B09902061, B09902054, B09902105

我們只做Find Similar這個task

N = 10000 = 有幾封 mail

$Q \approx 82000$ = 有幾個 Find Similarity Queries

q = $\frac{Q}{N} \approx 8$ = 對每封 mail 平均有幾筆 Find Similarity Queries

1. Data Structures & Algorighm

- 資料結構：二維陣列及一維陣列
- 演算法：離線建表/排序/Hash
- 原因：
  - 離線建表：因為算出兩兩similarity很花時間，所以先在本機算完，然後直接寫在code裡面。
  - 排序：排序query使得程式執行更快
  - Hash：因為原本的表太大，可以用類似hash的方式壓縮原本的表
- 時間複雜度$O(N^2 + QN)$
- 額外空間複雜度$O(N^2)$

基本做法

1. OJ 上 mail 的排列順序跟本地的相同，所以可以先離線建出兩兩 mail 間similarity的表。
2. 開一個二維陣列 $similar[N][N]$，$similar[i][j]$初始化為第i封mail與第j封mail的間的similarity。
3. 對於一個$query\{mid, threshold\}$，找出所有j滿足$similar[mid][j] > threshold$即可。

> 問題：每個浮點數因為精確度的關係，至少需要5個字元表示，整份檔案太大，無法上傳至OJ

優化一 將表格壓縮

1. 觀察發現每封email token set大小至多只有4000，於是我們可以用兩個字元來表示email兩兩的token交集大小，再利用 $|A \cup B| = |A| + |B| - |A \cap B|$求出兩兩的聯集與similarity。
   - 將數字轉成62進位，再用[a-z][A-Z][0-9]分別代表0~61，就可以用兩個字元表示一個整數了。
2. 不重複存(i, j)與(j, i)的similarity，進一步將表的大小壓到原本的一半。

> 成功將code大小壓縮至大約90MB

優化二 優化處理query的順序

- 由於query依序做會超時，因此我們先對同mid的query按照thereshold由小到大排序，在做threshold大的query時便可沿用前面的結果繼續篩選，可減少執行時間。

2. Cost Estimations of Queries

- 從字元陣列建similarity 表，時間及空間複雜度均需要$O(N^2)$
- 按threshold個別排序對於每封mail的query，如果用c語言的qsort排序，總共時間複雜度約為$O(Nq\lg q)$、空間複雜度為$O(Q)$
- 每次query檢查mid與其他mail的similarity，共有Q次，時間複雜度為$O(QN)$、空間複雜度為$O(1)$
- 整體的時間複雜度$O(QN)$，空間複雜度$O(N^2)$

3. Scheduling Strategy

- 策略：只做Find Similar
- 原因：Find Similar的分數遠大於其他task，又可以做很多次。
- 複雜度為$O(QN)\approx 8*10^8$可能會跑到8秒左右，但是扣除api.init()的時間其實只花了大約5秒，代表優化處理query的順序十分有效。

4. Additional Notes

離線建立similarity表格

1. 對所有mail建立一個陣列代表他的token set，此陣列要從小到大排序好，且不重複。
    - 利用c++的set可以將所有讀到的token排好且移除重複的。
2. 求出兩兩mail間token set的交集數量

```
int AND_size(int i, int j):
    string set1[N]=token_set[i]
    string set2[M]=token_set[j]
    int p1=0,p2=0,cnt=0
    while p1<N and p2<M :
        if set1[p1]<set2[p2] : p1++
        elif set1[p1]>set2[p2]: p2++
        else:
            p1++
            p2++
            cnt++
    return cnt
```

- **安排query順序在平均情況下的常數分析**
  - 假設threshold和similarity的分佈平均，且每封mail的query平均有q筆
  - 每次利用上一個thereshold得到的結果可以減少$\frac{1}{q}$筆
  - 只需要檢查$\frac{1}{q}+\frac{2}{q}+...+\frac{q}{q}=\frac{(q+1)}{2}$次，等於少看一半。